VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2015

Jiří Šimkovič

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Šimkovič**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: GIRITON Systems s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Gaura**

Konzultant bakalářské práce: Ing. Jan Řeš, MSc.

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA OSTRAVA Fakulta elektrotechniky a informatiky OSTRAVA 1

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7.5.2015

Šimbovič .....

Rád bych na tomto místě poděkoval firmě GIRITON Systems s.r.o a celému pracovnímu kolektivu za poskytnutí možnosti absolvování odborné praxe a také za jejich rady a vstřícnost během průběhu praxe. Dále pak panu Ing. Janu Gaurovi za udělené rady a připomínky při tvorbě této práce.

Abstrakt

Cílem této bakalářské práce je popsat průběh absolvování odborné praxe ve firmě GIRITON Systems s.r.o. V této firmě, která se zabývá tvorbou podnikových informačních systémů, jsem pracoval jako programátor.

V bakalářské práci uvedu nejprve seznámení s firmou GIRITON Systems s.r.o. a posléze se budu zabývat jednotlivými úkoly, které jsem během odborné praxe vykonával. Poté uvedu zhodnocení svých znalostí, nabytých během studia a praxe. Na závěr zhodnotím odbornou praxi jako celek.

Klíčová slova: GIRITON Systems s.r.o, JAVA, Vaadin

Abstract

The aim of this bachelor's thesis is to describe the process of absolving a professional practice in GIRITON Systems s.r.o. This company develops business Information Systems and I worked there as a programmer.

In this bachelor's thesis I will first write about the company itself and then I will proceed to describing particular tasks, I worked on during practice. After that I will evaluate knowledge gained during studies and practice. Lastly, I will assess the practice as a whole.

Keywords: GIRITON Systems s.r.o, JAVA, Vaadin

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
API	–	Application Programming Interface
CSS	–	Cascading Style Sheets
DOM	–	Document Object Model
GUI	–	Graphical User Interface
GWT	–	Google Web Toolkit
HTML	–	Hyper Text Markup Language
IS	–	Informační systém
REST	–	Representational State Transfer
RFID	–	Radio Frequency Identification
RMI	–	Remote Method Invocation
SASS	–	Syntactically Awesome Style Sheets
UML	–	Unified Modeling Language

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Pro vypracování bakálářské práce formou praxe jsem se rozhodl hlavně kvůli možnosti získání praktických zkušeností již během studia. Věřím, že získáné znalosti a dovednosti, nabyté během absolvování praxe, mi v budoucnu poskytnou více výhod než vypracování teoretické bakalářské práce. Ať už při dalším studiu, či na pracovním trhu.

V této práci nejdříve uvedu informace o firmě, ve které jsem praxi absolvoval, a mém pracovním zařazení v rámci firmy. V dalších kapitolách se budu postupně zabývat použitými technologiemi, kdy u každé uvedu úkol, který jsem s danou technologií zpracovával a výsledné řešení. Dále v práci provedu zhodnocení svých znalostí - jaké znalosti jsem získal při studiu a uplatnil je v průběhu praxe a jaké znalosti mi na praxi chyběly a musel jsem je doplnit. Závěr mé práce bude potom patřit dosaženým výsledkům v průběhu odborné praxe a jejímu celkovému zhodnocení.

2 Informace o praxi

2.1 Firma GIRITON Systems s.r.o

GIRITON Systems s.r.o. (dále jen GIRITON Systems) je mladá a dynamická firma, zabývající se tvorbou podnikových informačních systémů. Soustředí se na moderní cloudové systémy a jejich propojení s mobilními telefony a tablety.[1]

2.2 Informační systém GIRITON

IS GIRITON je nástroj pro sledování výroby a zakázek v reálném čase, řízení procesů a vyhodnocení docházky. IS GIRITON se skládá ze dvou hlavních součástí, kterými jsou **Docházka GIRITON** a **Řízení výroby**.

2.3 Docházka GIRITON

Docházka GIRITON umožňuje jednoduše a za minimální pořizovací náklady sledovat, evidovat a vyhodnocovat docházku zaměstnanců a to jak v terénu, tak uvnitř budov. Kromě klasických docházkových terminálů funguje také na tabletech a mobilních telefonech, které mohou k záznamům připojit GPS pozici nebo fotografii. Data jsou předávána cloudové aplikaci, ve které lze v intuitivním rozhraní vše přehledně spravovat a v závislosti na oprávnění uživatelů provádět exporty nebo nastavovat automatické úlohy.[2]

2.4 Řízení výroby

IS GIRITON je pokrokový systém na řízení a sledování procesů v malých a středních firmách výrobního a zpracovatelského průmyslu. Procesy a události jsou monitorovány v reálném čase pomocí levných tabletů a mobilních telefonů, nasazených přímo v podniku. Pomocí čárových kódů a RFID tento systém zaznamenává, co který zaměstnanec v danou chvíli dělá, na které zakázce a s jakým materiálem pracuje. Tím je sledována nejen přesná aktuální rozpracovanost, ale i přesné stavy skladů. Ať už jde o výrobu zakázkovou, na sklad či výrobu kombinovanou, vždy je k dispozici také přehled o stavu objednávek.[3]

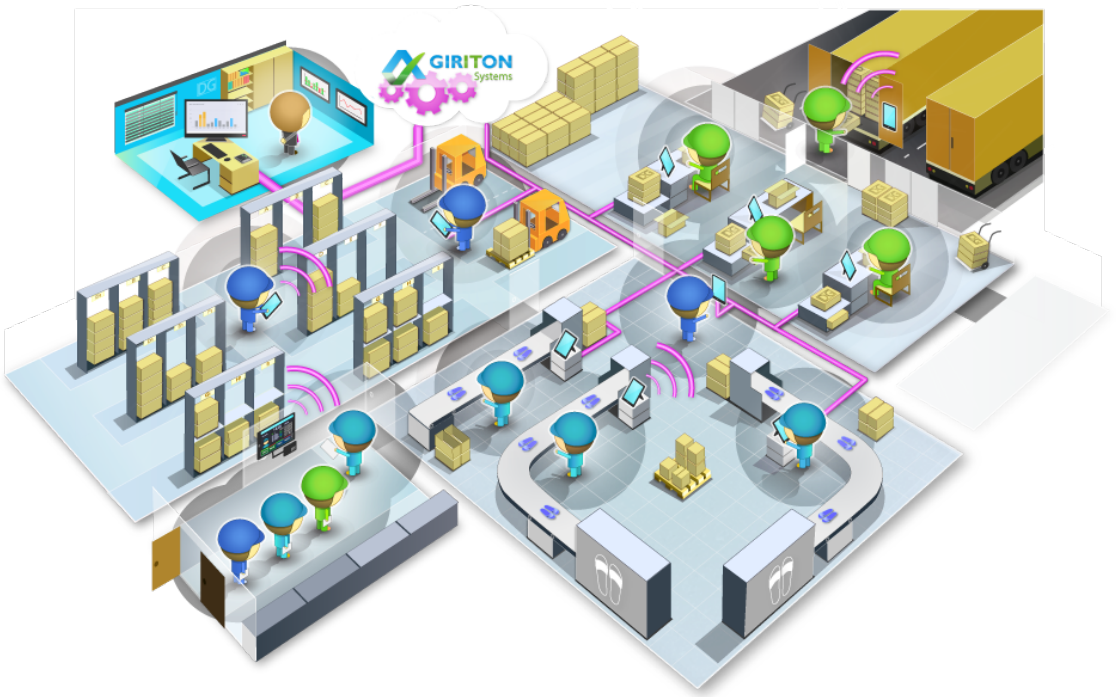
2.5 Pracovní zařazení

Do firmy jsem nastoupil na pozici Java programátora. Jelikož jsem ale měl předchozí znalosti HTML, CSS a dalších webových technologií, začal jsem tvorbou interaktivního obrázku s informacemi o firmě pomocí obrázkových map. Po splnění tohoto úkolu jsem se již začal seznamovat se samotnou docházkovou aplikací prostřednictvím programování integračních a jednotkových testů. Jelikož již byla aplikace v té době značně rozsáhlá a obsahovala mnoho různých oblastí, pomohlo mi psaní jednotlivých testů lépe se zorientovat a získat přehled v samotném kódu aplikace. Ten jsem využil při práci na samotné aplikaci, kdy jsem upravoval a vytvářel různé GUI elementy. Během praxe jsem také pokračoval v psaní jak integračních, tak jednotkových testů pro nově přidané funkce. Dále

jsem pracoval na vytvoření REST API k docházkové aplikaci, která byla vytvářena za účelem propojení aplikace se zařízeními s operačním systémem iOS.

3 HTML Obrázkové mapy

Jako první úkol mi bylo ve firmě přiděleno prozkoumat a následně naimplementovat možnost využití obrázkových map pro zobrazení obrázku (Obrázek 1) a příslušných informací po najetí na jeho různé části. Na tomto obrázku se totiž nacházely různé oblasti působnosti firmy GIRITON Systems a smyslem bylo zobrazit krátký popis s dalšími informacemi pro každou oblast. Při práci na tomto úkolu a při psaní této kapitoly jsem čerpal z webové stránky `W3Schools`[4].



Obrázek 1: Původní obrázek pro obrázkovou mapu

3.1 Popis technologie

Obrázkové mapy v HTML jsou běžné obrázky, specifikované v `<img>` tagu, ke kterým jsou připojeny informace o různých oblastech. Tyto oblasti jsou definovány zadáním jejich tvaru a pozice. Ve výsledku lze tedy definovat např. obrázek České republiky, kde jednotlivé oblasti budou představovat různé kraje. Díky tomuto je možné odlišit kliknutí na jednotlivé kraje v obrázku a zobrazit příslušné informace ke každému kraji.

Samotné mapy se v HTML definují pomocí párového tagu `<map>` s atributem „name“, který značí název mapy. Tato mapa se pak k obrázku připojí pomocí atributu „usemap“, definovaného u tagu <img>. Dovnitř tagu <map> se pak definují jednotlivé oblasti pomocí tagu <area>. Atributy tohoto tagu určují mj. pozici oblasti, její tvar a umístění, na

které bude tato oblast odkazovat. Ve výpisu kódu (Výpis 1) je k vidění základní použití popsaných tagů.

```
<img src="republika.png" width="145" height="126" alt="Republika" usemap="#republikaMap">

<map name="republikaMap">
  <area shape="rect" coords="0,0,82,126" href="moravskoslezsky.html" alt="Moravskoslezsky␣
      kraj">
  <area shape="circle" coords="90,58,3" href="stredocesky.html" alt="Stredocesky␣kraj">
  <area shape="poly" coords="124,58,8" href="jihocesky.html" alt="Jihocesky␣kraj">
</map>
```

Výpis 1: Ukázka použití obrázkové mapy

Při tomto úkolu jsem také použil HTML element `canvas`, který slouží pro vykreslování grafiky do webového prohlížeče.

3.2 Popis zadaného úkolu

Obrázek, se kterým jsem pracoval, představuje podnik rozdělený na různé části, jako vstupní místnost, kde se dá využít firmou nabízené sledování docházky, nebo výrobna, pro kterou firma nabízí monitoring výroby. Kromě těchto dvou se na obrázku nachází i další části. Každá z těchto částí měla být odlišitelná od ostatních a při najetí myši na konkrétní část se mělo zobrazit informační okno s dodatečnými informacemi. V neposlední řadě se měla při přejetí myši nad některou částí daná část zesvětlit a okolní části ztmavit.

3.3 Popis řešení

Řešení tohoto úkolu jsem začal zjišťováním možných alternativ. Po zjištění, že HTML obrázkové mapy jsou pro tento úkol ideální, jsem si o nich zjistil informace, které jsou zhruba uvedeny v kapitole 3.1.

Prvním problémem se ukázalo definování oblastí v obrázku, jelikož jednotlivé části byly různorodé mnohoúhelníky. Tento úkol mi značně usnadnila webová stránka `http://www.image-maps.com/`. Na této stránce byl online nástroj pro tvorbu oblastí v obrázku. Po nahrání obrázku do online nástroje jsem v něm mohl umisťovat body, které určily jednotlivé mnohoúhelníky. Vygenerované oblasti jsem poté použil při řešení tohoto úkolu. Výsledný HTML kód lze vidět ve výpisu kódu (Výpis 2).

```
<img width="950" height="600" id="immap" src="img/podnik.png" usemap="#pmap" alt="Podnik"/
    >
        <div id="podnik_popup_outer">
            <div id="podnik_popup">
                <h1>Popup text</h1>
                <p>Tady je popup text!</p>
            </div>
        </div>
        <br>
        <map name="pmap">
            <area  alt="Vyroba" title="" href="#" shape="poly" coords="
                590,576,813,378,538,213,314,345" onmouseover="myHover(this);"
                onmouseout="myLeave();" />
            <area  alt="Dochazka" title="" href="#" shape="poly" coords="souradnice"
                onmouseover="myHover(this);" onmouseout="myLeave();" />
            dalsi  <area> tagy ....
        </map>
```

Výpis 2: HTML kód obrázkové mapy

Výpis kódu také obsahuje prvek `<div id="podnik_popup_outer">` a další vnořené prvky. Tyto prvky jsou použity pro zobrazení informačního okna. Zobrazování bylo dosaženo pomocí skriptovacího jazyku JavaScript a knihovny jQuery. Starají se o to metody `myHover` a `myLeave`, které jsou uvedeny v atributech `onmouseover` a `onmouseout` tagu `<area>` u jednotlivých oblastí. Kód těchto metod je k vidění ve výpisu (Výpis 3).

```
function myHover(element)
{
    var coordStr = element.getAttribute('coords');
    var  alt=element.getAttribute('alt');

    setPopupPosition(alt);
    setPopupText(alt);
    $('#podnik_popup_outer').show();
    drawPoly(coordStr);
}
function myLeave()
{
    hdc.clearRect(0, 0, can.width, can.height);
    $('#podnik_popup_outer').hide();
}
```

Výpis 3: Kód najetí a odjetí myši

Funkce `setPopupPosition(alt)` a `setPopupText(alt)` podle atributu `alt` elementu přiřadí prvku informačního okna text a pozici. Poté se zobrazí prvek s informačním oknem pomocí metody `show()` a nakonec se vykreslí ztmavení a zesvětlení obrázku. Ve funkci `myLeave()` se zruší ztmavení a zesvětlení a schová se prvek s informačním oknem. Tyto funkce jsou poměrně jednoduché, složitější funkcionalita je implementována ve funkci `drawPoly(coords)`, která se stará o ztmavení a zesvětlení oblastí

na obrázku. Její kód je ve výpisu (Výpis4). V tomto výpisu jsou použity také proměnné `can`, reprezentující canvas umístěný přesně na obrázek a `hdc`, což je jeho context pro kreslení. Funkce `drawPoly(coords)` získá souřadnice jednotlivých bodů mnohoúhelníku, ohraňujícího některou z částí. Poté si vytvoří černý obrázek s nastavenou průhledností a s rozměry původního obrázku. Dále se z daných souřadnic vytvoří v černém obrázku mnohoúhelník a vybarví se bílou barvou (také s průhledností). Díky parametru `globalCompositeOperation` ‚nastavenému na 'xor', dojde k vyrušení bílé oblasti s černou oblastí pod ní a zůstane tak v tomto obrázku prázdná „díra" přesně pro danou část z původního obrázku. Vytvořený obrázek poté překryje obrázek původní a díky tomu je docíleno výsledného efektu. Tento efekt můžete videt na obrázku (Obrázek 2).

Obrázek 2: Hotový obrázek s obrázkovou mapou

```
function drawPoly(coOrdStr)
{
    var mCoords = coOrdStr.split(',') ;

     var maskCanvas = document.createElement('canvas');

     maskCanvas.width = can.width;
     maskCanvas.height = can.height;
     var maskCtx = maskCanvas.getContext('2d');

     maskCtx. fillStyle  = "rgba(0,0,0,0.7)" ;
     maskCtx.fillRect(0,  0,  maskCanvas.width, maskCanvas.height);
     maskCtx.globalCompositeOperation = 'xor';
     maskCtx. fillStyle  = "rgba(255,255,255,0.9)";

     var j ,  m;
     m = mCoords.length;
     maskCtx.beginPath();
     maskCtx.moveTo(mCoords[0], mCoords[1]);
     for  (j  = 2;  j  < m; j  += 2)
     {
          maskCtx.lineTo(mCoords[j], mCoords[j + 1]);
     }
     maskCtx.lineTo(mCoords[0], mCoords[1]);
     maskCtx.closePath();
     maskCtx.stroke();
     maskCtx. fill () ;

     hdc.drawImage(maskCanvas, 0, 0);
}
```

Výpis 4: Kód ztmavení a zesvětlení na obrázku

4 Integrační testování

Mým druhým úkolem během bakalářské praxe bylo testování webové aplikace. Ze začátku jsem měl na starost psaní integračních testů pomocí nástroje Selenium, dále pak psaní jednotkových testů ve frameworku jUnit. V této kapitole se však budu zabývat pouze psaním integračních testů. Při popisu technologie využívám znalosti získané z webových stránek projektu Selenium [5]. Ve výpisu jednotlivých používaných metod pak volně používám programátorskou dokumentaci [6].

4.1 Popis technologie

Selenium je skupina nástrojů, které mají za úkol automatizovat webové prohlížeče. Selenium dává uživateli vzdálenou kontrolu nad webovým prohlížečem a umožňuje mu tak napodobovat chování běžného uživatele. Selenium 2.0 poskytuje kontrolu nad prohlížeči díky rozhraní `WebDriver`. Pomocí tohoto rozhraní a jeho implementací se dají ovládat prohlížeče jako Firefox, Chrome, Internet Explorer, Safari a Opera. Ovládání prohlížeče Firefox je realizováno pomocí `FirefoxDriver`, který je součástí Selenia a jeho použití je proto asi nejjednodušší, jelikož se nemusí k testovacímu projektu připojovat další závislosti. Pro ovládání prohlížeče Chrome by například bylo potřeba stáhnout nebo jinak k projektu připojit `ChromeDriver`. Selenium také podporuje vzdálené testování prohlížeče, běžícího na jiném PC pomocí `RemoteWebDriver`.

Dalším důležitým objektem je objekt typu `WebElement`. Rozhraní `WebElement` reprezentuje HTML element a poskytuje množinu operací, které mohou být s tímto objektem prováděny.

Po inicializaci objektu `WebDriver` se otevře nová instance prohlížeče, ve které se následně provádí samotná manipulace s prohlížečem a testování. Na tento objekt pak můžeme volat metody, které nám poskytnou informace o prohlížeči nebo metody, které nám poskytnou elementy k manipulaci.

Metody, které jsem používal nejvíce, byly tyto:

get(String url)
: Otevře ve webovém prohlížeči určenou stránku.

findElement(By by)
: Vrátí první výskyt elementu podle jeho id, třídy, nebo podle XPath.

findElements(By by)
: Vrátí všechny elementy, které na stránce nalezne, podle dané metody.

quit()
: Ukončí driver a zavře všechna jeho okna.

Typické použití samotného objektu `WebDriver` tedy spočívalo především v otevření stránky, nalezení potřebných elementů a poté jeho ukončení. Před jeho ukončením jsem pracoval s jednotlivými elementy pomocí následujících metod:

getText()
Vrátí viditelný text uvnitř tohoto elementu.

click()
Klikne na tento element.

sendKeys(CharSequence... keysToSend)
Simuluje psaní textu do elementu.

Získávání jednotlivých elementů čistě pomocí objektu `WebDriver` je ale problematické, jelikož se často stává, že se daný element ještě nenačetl a v tomto případě by test nebyl úspěšně dokončen. Tento problém řeší další třída, a to `WebDriverWait`. Tato třída má metodu `until(com.google.common.base.Predicate<T> isTrue)` a funguje nejlépe v kombinaci s některou ze statických metod třídy `ExpectedConditions`, která je součástí Selenia. Některé z používaných metod této třídy jsou uvedeny níže.

visibilityOfElementLocated(By locator)
Zkontroluje, zda je specifikovaný element přítomen v DOM a je viditelný.

elementToBeClickable(By locator)
Zkontroluje, zda je specifikovaný element viditelný a dá se na něj kliknout.

Typické použití tříd `WebDriverWait` a `ExpectedConditions` je před vybráním elementu z DOM pomocí `WebDriver` a jeho metody `findElementBy(By locator)`. Toto použití je demonstrováno v následující ukázce.

```
WebDriver driver = new FirefoxDriver();
driver.get("http :// www.adresa.cz");
WebDriverWait wait=new WebDriverWait(driver, 10);
wait. until (ExpectedConditions.visibilityOfElementLocated(By.id("button"));
driver.findElement(By.id("button)). click ();
```

Výpis 5: Typické použití Selenium

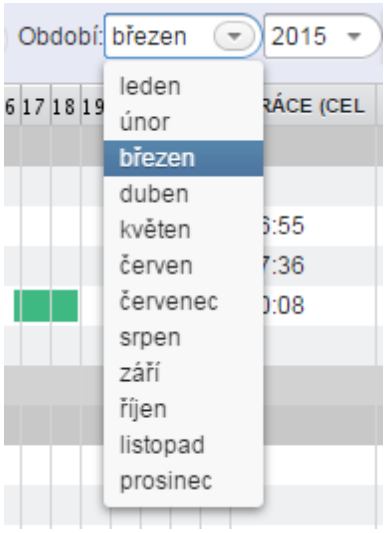
4.2 Popis zadaného úkolu

S nástrojem Selenium jsem pracoval jednak při úpravě stávajících testů, tak při vytváření nových testů podle zadaného scénáře. V rámci jednoho scénáře jsem měl za úkol otestovat zobrazení docházkových dat v aplikaci. Během tohoto testu jsem tedy musel vytvořit nového uživatele, přidat mu data docházky, přihlásit se do webové aplikace a zkontrolovat zobrazení dat v příslušné části aplikace.

4.3 Popis řešení

Pro vytvoření tohoto testu jsem potřeboval nejen znalost nástrojů Selenium, ale i povědomí o existenci a způsobech použití různých RMI tříd. Pomocí těchto tříd jsem nastavoval data pro test, jelikož zadávat je pomocí Selenia by bylo velice zdlouhavé a zbytečné. Je také dobrým pravidlem psát Selenium testy jako co nejmenší jednotky, jelikož je jejich

provádění vcelku nákladné. Jednotlivé RMI třídy jsem tedy použil pro vytvoření uživatele a naplnění jeho účtu docházkovými daty. Přesný způsob je uveden v příloze ve výpisu kódu (Výpis 6). Zbylé úkony jako přihlášení, přechod na příslušnou agendu v aplikaci a samotnou kontrolu zobrazení dat jsem již provedl pomocí Selenium. Ukázka je zkrácená a je v ní vynecháno několik částí. Metoda přihlášení a vybrání příslušné agendy není ukázána, jelikož jsem použil již existující řešení z testů jiných. Musel jsem ale implementovat vybrání měsíce a roku ze skupiny rozbalovacích seznamů z obrázku (Obrázek 3). Celý kód testu je v ukázce umístěn do jediné metody, což je pouze pro zjednodušení, jelikož ve skutečnosti je rozdělen na více míst kvůli znovupoužitelnosti kódu.



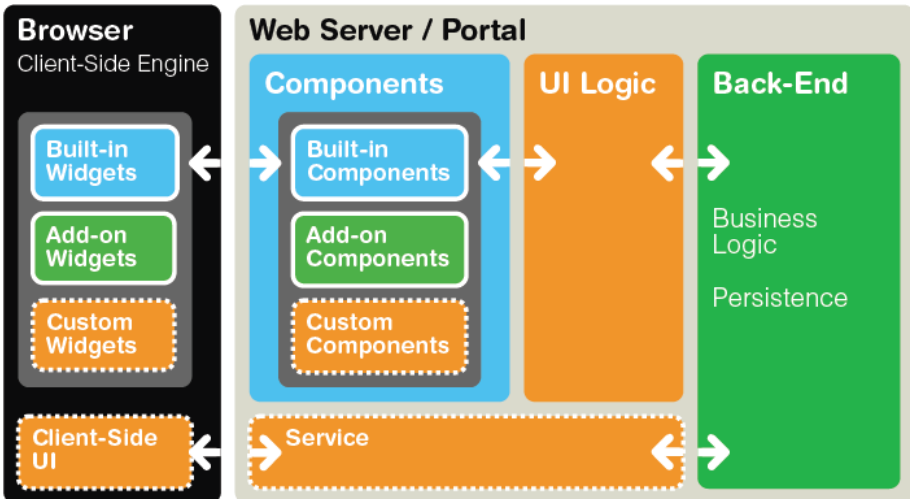
Obrázek 3: Skupina rozbalovacích seznamů

5 Vaadin

V tomto frameworku byl napsán IS GIRITON a součástí mé praxe bylo se s ním dobře seznámit, abych byl schopen v IS provádět potřebné úpravy a přidávat novou funkcionalitu. V následující kapitole uvedu základní popis Vaadin frameworku a na úkolech, obdržených během praxe, uvedu také příklad použití. Při popisu technologie a používaných metod používám informace z knihy `Book of Vaadin`[7].

5.1 Popis technologie

Vaadin je framework určený pro psaní webových aplikací. Kód je psán v jazyce Java a poté pomocí GWT překládán do JavaScriptu, který je následně interpretován ve webovém prohlížeči. Programátor nemusí mít žádné znalosti HTML, CSS ani JavaScriptu, aby vytvořil bohatou internetovou aplikaci. Architektura aplikace, vytvořené ve Vaadinu, je na obrázku (Obrázek 4).

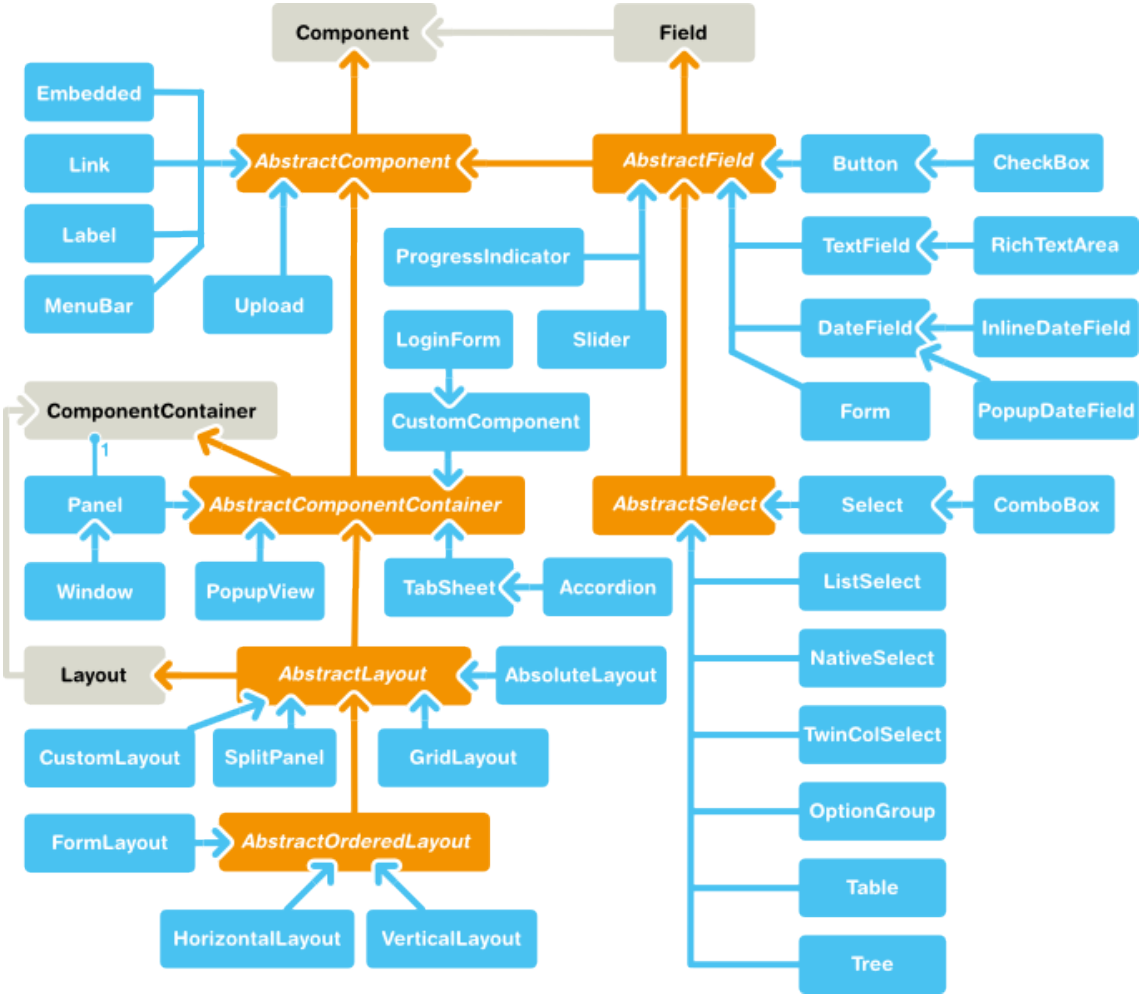


Obrázek 4: Architektura Vaadin aplikace [7]

Ve webovém prohlížeči běží jen tenká vrstva uživatelského rozhraní, která je vykreslována pomocí JavaScriptu. Všechny změny jsou zpracovávány serverem a jsou na něj zasílány pomocí AJAXu. Různé události, jako jsou stisknutí tlačítka, vybrání položky ze seznamu nebo odeslání formuláře, jsou tedy zaslány a zpracovány na serveru pomocí odpovídajících posluchačů, definovaných v Java kódu.

Vaadin obsahuje velké množství prvků uživatelského rozhraní - komponent a také umožňuje definovat vlastní prvky. Přehled těchto komponent je na obrázku (Obrázek 5). Na tomto obrázku jsou jednotlivé třídy zobrazeny modře, rozhraní šedě a abstraktní třídy oranžově. Komponenty, které rozšiřují rozhraní `ComponentContainer`, mohou v sobě obsahovat i další komponenty. Z tohoto pohledu se při tvorbě uživatelských rozhraní využívají hlavně komponenty, dědící z třídy `AbstractLayout`. Tyto komponenty

umožňují programátorovi manipulovat s uživatelským rozhraním a uspořádat jednotlivé prvky dle libosti. Tvorba uživatelského rozhraní pak ve své nejjednodušší podobě spočívá ve vytvoření jednotlivých prvků a umístění do příslušných komponent, dědících ze třídy `AbstractLayout`. Po nasazení aplikace se pak ve webovém prohlížeči vykreslí vše potřebné a programátor se může nadále věnovat zbylé logice aplikace. Programátor ale také může využít třídu `CustomLayout`, která rozvržení jednotlivých komponent přebírá z poskytnuté HTML šablony.



Obrázek 5: Uživatelské komponenty Vaadinu [7]

Vzhled uživatelského rozhraní je řízen tématy, která jsou tvořena pomocí SASS, nebo CSS. Toto odděluje logiku použití jednotlivých komponent od jejich vzhledu a umožňuje tak oddělený vývoj těchto dvou částí. Vaadin nabízí několik základních témat, která se dají použít ať už jako celkové řešení nebo mohou poskytnout základ pro vytvoření vlastního téma.

5.2 Popis zadaného úkolu

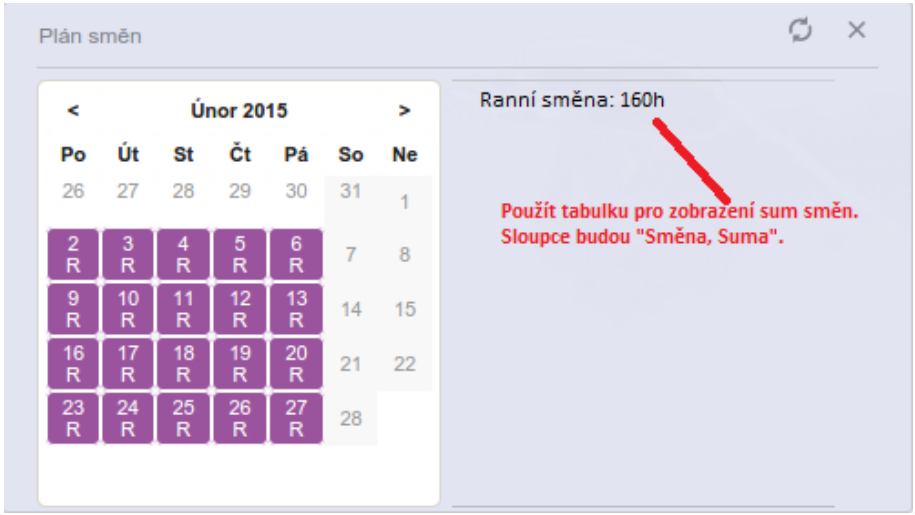
Po seznámení s frameworkem Vaadin jsem postupně začal pracovat na samotném IS GI-RITON. Zprvu jsem měl na starost drobné úpravy různých částí uživatelského prostředí, při kterých šlo většinou o změnu rozložení a přidání/úpravu ovládacích prvků. Takovýchto drobných úprav jsem za dobu praxe udělal velké množství. V této kapitole tedy uvedu pouze dva příklady, a to úpravy okolo widgetu „Plán směn“ a úpravy okna s podrobnostmi docházkového záznamu. Při těchto drobných úpravách jsem získal více zkušeností s Vaadinem a později jsem dostal za úkol i tvorbu vlastní nové části IS.

Nová část měla být kontrola fotografií. Docházkové terminály, používané pro zadávání docházky, totiž mohou být nastaveny tak, že při každém záznamu pořídí i fotografii. Tyto fotografie tedy bylo třeba zobrazit uživateli pro kontrolu, zda se zaměstnanci například nehlásí do práce za někoho jiného. Při prohlížení docházky mělo být osobám s příslušným oprávněním zobrazeno tlačítko, po jehož stisknutí si mohli vybrat konkrétní osoby pro kontrolu fotografií. Po vybrání osob se již měly v IS zobrazit všechny vybrané osoby spolu se všemi jejich fotkami a informacemi určující každou z fotek.

5.3 Popis řešení

5.3.1 Widget „Plán směn“

U tohoto widgetu jsem měl za úkol upravit zobrazení sum směn na domovské stránce IS. Tento widget obsahoval kalendář směn, který zabíral přibližně polovinu šířky widgetu, a shrnutí sum spolu s názvem směny, ke kterému se suma vztahuje. Zobrazení sum mělo být upraveno do přehledné tabulky. Původní podobu spolu s textovým zadáním můžete vidět na obrázku (Obrázek 6).



Obrázek 6: Původní podoba widgetu

Úpravu tohoto widgetu jsem realizoval tabulkou, kterou ve Vaadinu představuje třída `Table`. U vytvořené tabulky jsem využil tyto metody:

addContainerProperty(Object propertyId, Class<?> type, Object defaultValue)
Přidá do tabulky nový sloupec.

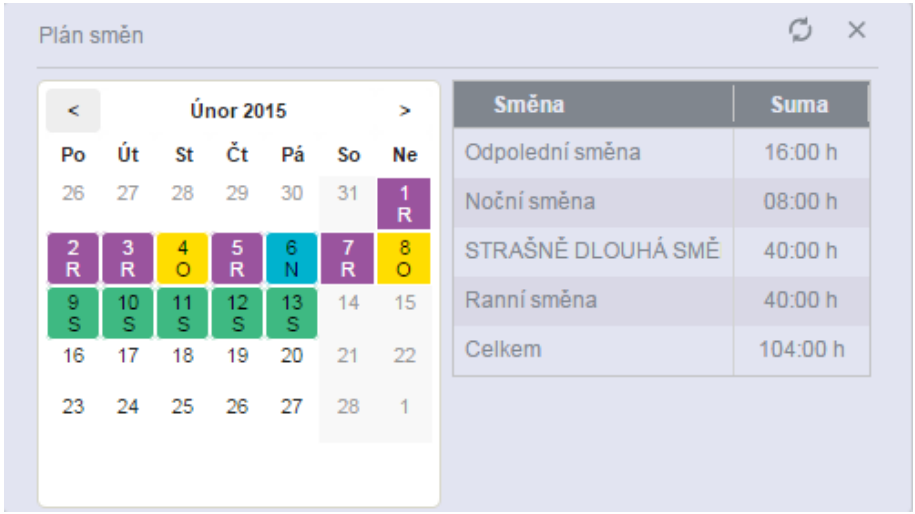
setWidth(float width, Sizeable.Unit unit)
Nastaví šířku tabulky.

setColumnExpandRatio(Object propertyId, float expandRatio)
Nastavuje poměr, ve kterém si sloupce rozdělí prostor v tabulce.

setColumnAlignment(Object propertyId, Table.Align alignment)
Nastaví zarovnání příslušného sloupce.

addItem(Object[] cells, Object propertyId)
Jedna z metod, která přidá do tabulky nový řádek.

Výsledná podoba widgetu je na obrázku (Obrázek 7).



Obrázek 7: Výsledná podoba widgetu

5.3.2 Podrobnosti záznamu docházky

Při prohlížení docházky bylo v IS GIRITON možné zobrazit i okno s podrobnostmi docházkového záznamu, které obsahovalo dodatečné informace o daném záznamu. Do tohoto okna jsem měl za úkol přidat nové prvky a také upravit jeho vzhled. Původní podobu okna s detailem záznamu a zadání změn můžete vidět na obrázku (Obrázek 8).

Obrázek 8: Původní podoba okna s detailem záznamu

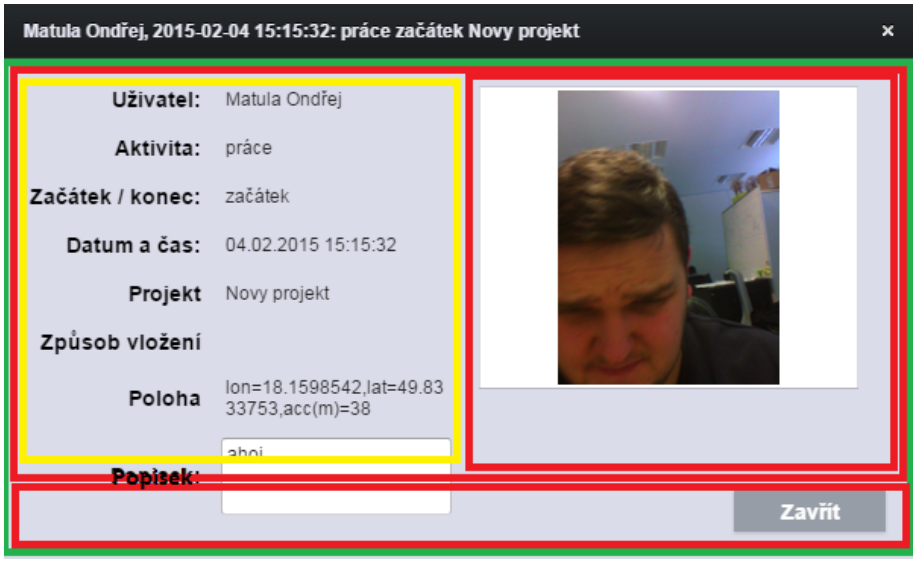
Při úpravě tohoto okna bylo potřeba změnit rozložení jednotlivých částí. To se ve Vaadinu obecně provádí pomocí komponent, jejichž úkolem je uchovávat další prvky. Jedná se o již zmíněné komponenty, dědící ze třídy `AbstractLayout` a konkrétně jde o tyto:

HorizontalLayout
: Uspořádá vložené komponenty vedle sebe.

VerticalLayout
: Uspořádá vložené komponenty pod sebe.

FormLayout
: Uspořádá vložené komponenty do dvou sloupců. V prvním sloupci je popis komponenty a ve druhém pak komponenta samotná.

Všechna tato rozvržení jsem použil při úpravě tohoto okna. Výsledné okno s vyznačenými rozvrženími, kde červená barva představuje `HorizontalLayout`, zelená barva `VerticalLayout` a žlutá barva `FormLayout`, můžete vidět na obrázku (Obrázek 9). Toto okno dále obsahuje komponenty: `TextField`, `TextArea`, `Button` a `PicturePanel`.

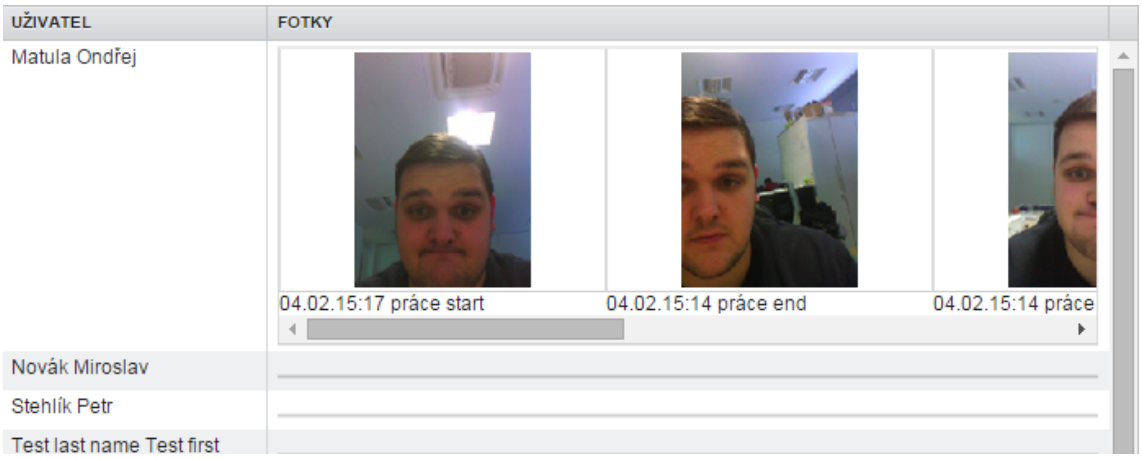


Obrázek 9: Výsledná podoba okna s detailem záznamu

5.3.3 Kontrola fotografií

Vzhledem k povaze zobrazovaných dat jsem si i pro tento úkol vybral zobrazení pomocí tabulky. Jedním sloupcem bylo jméno uživatele a druhým všechny zobrazené fotky. Do druhého sloupce jsem ale potřeboval umístit velké množství dat. Ke každé fotce měly být zobrazeny informace, sloužící k její identifikaci, a fotek mohlo být velké množství, které se do tabulky nemuselo vejít. Bylo tedy třeba zobrazovat i posuvník pro projíždění fotek.

Zobrazení druhého sloupce mi usnadnil fakt, že buňka tabulky mohla obsahovat i jiné Vaadin komponenty. Při volání metody `addContainerProperty(..)` jsem tedy jako typ dat ve sloupci zadal třídu `Panel`. Ačkoli by se na první pohled mohlo zdát, že by pro zobrazení všech fotek vedle sebe v tabulce byl ideální `HorizontalLayout`, tato třída by nevyvolala zobrazení posuvníku při přemíře dat v tabulce. Tohoto efektu se dalo dosáhnout pouze použitím třídy `Panel`, do které se jako obsah nastavil `HorizontalLayout`. Ten pak obsahoval `VerticalLayout` pro každý záznam s fotkou, ve kterém byl `PicturePanel` obsahující fotku a `Label` s informacemi o příslušném záznamu. Celou tabulku můžete vidět na obrázku (Obrázek 10), kód se pak nachází v příloze ve výpisu kódu (Výpis 7).



Obrázek 10: Kontrola fotografií

6 Zhodnocení znalostí

6.1 Uplatněné teoretické a praktické znalosti

Během odborné praxe jsem využil hlavně znalosti získané v předmětech zabývajících se programováním. Hlavním přínosem pro mne v tomto ohledu byl předmět *Programovací jazyky I*, jelikož se zabýval jazykem Java, který jsem používal téměř celou praxi. Se znalostí jazyka Java mi také pomohl zahraniční studijní pobyt v rámci programu Erasmus. V rámci tohoto pobytu jsem totiž pracoval ve firmě, která vyvíjela webové aplikace převážně v jazyce Java. Tento pobyt mne také připravil na práci ve firmě a v týmu, což mi značně usnadnilo přechod do pracovního prostředí firmy GIRITON Systems.

Z dalších předmětů mi byly přínosné předměty *Algoritmy I* a *Algoritmy II*. Přestože během jejich výuky nebyl využíván jazyk Java, naučily mne tyto předměty základy programátorských dovedností. V neposlední řadě mi při seznamování se s frameworkem Vaadin pomohl předmět *Úvod do softwarového inženýrství*, díky němuž jsem se orientoval v jazyce UML.

6.2 Scházející znalosti a dovednosti

Při praxi mi nejvíce chyběly znalosti frameworků, používaných ve firmě. Musel jsem prostudovat frameworky Vaadin, Selenium a také framework Hibernate, který se v IS GIRITON používá pro mapování objektového modelu na relační databáze. S objektově relačním mapováním jsem se během studia setkal v předmětu *Databázové a informační systémy*, ve kterém šlo ale jen o vytvoření vlastní implementace. To mi sice poskytlo povědomí o problematice, ale práci s frameworkem Hibernate jsem musel dostudovat.

7 Celkové zhodnocení

Absolvování odborné praxe pro mne bylo jednoznačně velkým přínosem. Vyzkoušel jsem si skutečnou práci ve firmě, včetně spolupráce v týmu, pobytu v open space kanceláři a vývoje reálné aplikace pro reálné zákazníky. Kromě toho jsem si vyzkoušel práci s frameworky a technologiemi, se kterými bych se při studiu nesetkal.

Dále jsem měl možnost porovnat své dosavadně nabyté zkušenosti s tím, co je potřeba při práci ve skutečné firmě. Zjistil jsem, že ač mi studium na univerzitě dalo základní znalosti široké škály oborů, pro praxi je potřeba tyto znalosti více prohloubit.

Závěrem bych chtěl říci, že kdybych se rozhodoval znovu, volil bych opět vypracování bakalářské práce touto formou a zároveň bych to doporučil každému studentovi, který tuto možnost zvažuje.

8 Reference

[1] GIRITON Systems s.r.o. [online]. [cit. 2015-03-01]. Dostupné z: http://www.giriton.cz/cz/index

[2] GIRITON Systems s.r.o. Docházka [online]. [cit. 2015-03-01]. Dostupné z: http://dochazka.giriton.cz/cz/index.html

[3] GIRITON Systems s.r.o. Řízení výroby [online]. [cit. 2015-03-01]. Dostupné z: http://www.giriton.cz/cz/rizeni-vyroby

[4] W3Schools [online]. [cit. 2015-04-25]. Dostupné z: http://www.w3schools.com/

[5] Selenium [online]. [cit. 2015-04-25]. Dostupné z: http://docs.seleniumhq.org/

[6] Selenium API dokumentace [online]. [cit. 2015-04-25]. Dostupné z: https://selenium.googlecode.com/svn/trunk/docs/api/java/overview-summary.html

[7] Book of Vaadin [online]. [cit. 2015-03-01]. Dostupné z: https://vaadin.com/book/-/page/intro.html

A Zdrojové kódy

```
@Test
public void testSeptemberPresence() throws Exception {
FirefoxDriver driver;
WebDriverWait wait;
// RMI pro pristup do databaze
DatabaseBasicRmi rmi;
// RMI pro dochazku
RmiPresenceWorker presence;
// Vynechan kod inicializace promennych
// Vynechan kod pridani uzivatele pomoci DatabaseBasicRmi
// Vynechan kod pridani zaznamu dochazky pomoci RmiPresenceWorker
// Testovani spravneho nahrani testovacich dat
Person tempPerson;
tempPerson = (Person) rmi.executeJpqlQuery("Select␣ent␣from␣" + Person.class.getSimpleName()
    + "␣ent␣where␣number=:n", Collections.singletonMap("n", PERSON_NUMBER), Boolean.
    FALSE).get(0);
Assert.assertNotNull("Zamestnanec␣neexistuje", tempPerson);
List<PresenceEntry> entries = rmi.executeJpqlQuery("Select␣ent␣FROM␣" + PresenceEntry.class.
    getSimpleName() + "␣ent", null, Boolean.FALSE);
Assert.assertNotEquals("Neexistuji␣zaznamy␣dochazky", entries.size(), 0);

String workCleanXPATH = String.format("//div[@id='table-dochazka-sumy']/div/div/table/tbody/tr[
    contains(.,'prace')]/td[3]/div");
String workTotalXPATH = String.format("//div[@id='table-dochazka-sumy']/div/div/table/tbody/tr[
    contains(.,'Prace␣(celkem)')]/td[3]/div");
// Vynechan kod dalsich XPATHs
// Vynechan kod prihlaseni do aplikace a vybrani agendy s dochazkou
// ziskani rozbalivaciho seznamu s mesicem
WebElement monthBtn = (WebElement)fluentWaiter.until(new Function<WebDriver,WebElement>()
    {
@Override
public WebElement apply(WebDriver driver) {
return driver.findElement(By.xpath("//div[@id='dochazka-panel-date-moyepicker']//div[@class='
    v-slot'][1]/div/div[@class='v-filterselect-button']"));
}
});
// XPath ukazujici na odpovidajici mesic
String monthXPATH = String.format("//div[@id='VAADIN_COMBOBOX_OPTIONLIST']/div/div/table/
    tbody/tr[contains(.,'%s')]/td", "zari");
// kliknuti na rozbalovaci seznam
monthBtn.click();
// vybrani prislusneho mesice
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(monthXPATH)));
driver.findElement(By.xpath(monthXPATH)).click();
// Vynechan kod vybrani roku, ktery je temer identicky s vybranim mesice
// vyckani na obnoveni sum, ktere se provadi asynchronne
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//div[@id='table-dochazka-
    sumy']/div/div/table/tbody/tr[contains(.,'prace')]/td[3]/div")));
// porovnani hodnot z tabulky sum s ocekavanou skutecnosti
Assert.assertEquals("Suma␣ciste␣prace␣neodpovida", "20:00", driver.findElement(By.xpath(
    workCleanXPATH)).getText());
```

```
Assert.assertEquals("Suma␣prace␣celkem␣neodpovida", "47:00", driver.findElement(By.xpath(
    workTotalXPATH)).getText());
// Vynechan kod pro porovnani ostatnich hodnot a ukonceni testu

}
```

Výpis 6: Integrační test pro testování docházky

```
Table photosTable = new Table();
List<Person> persons=/* Vybrane osoby;

photosTable.setSizeFull();
photosTable.addContainerProperty("Uzivatel", Label.class, null);
photosTable.addContainerProperty("Fotky", Panel.class, null);
photosTable.setColumnWidth("Uzivatel", 150);
photosTable.setColumnExpandRatio("Fotky", 1);


for (Person person : persons) {
Label l = new Label(person.toStringGui());
Panel panel = new Panel();
if (/* Pokud ma person zaznamy s fotkou*/) {
HorizontalLayout hl = new HorizontalLayout();
for (PresenceEntry entry : /*Pro kazdy zaznam s fotkou*/) {
PicturePanel picture = picturesMap.get(entry.getPersonPhoto().getId());
VerticalLayout layEntry = new VerticalLayout();
layEntry.setWidth("200px");
picture.setHeight("150px");
layEntry.addComponent(picture);
l.setHeight("20px");
layEntry.addComponent(new Label(entry.getDateValid().toString(DateUtil.dayMonthFormater) +
    entry.getTimeOfEntry().toString(DateUtil.localTimeNoSecsFormater) + "␣" + entry.getActivity().
    getName() + "␣" + entry.getStartOrEnd().name()));
hl.addComponent(layEntry);
}
panel.setContent(hl);
}
Object itemId = photosTable.addItem();
Item item = photosTable.getItem(itemId);
item.getItemProperty("Uzivatel").setValue(l);
item.getItemProperty("Fotky").setValue(panel);
}
```

Výpis 7: Kód kontroly fotografií